Chapter 6

Link layer : data transfer between neighboring Network elements (ethernet, wifi, PPP)

↳ frame

↳ link layer transfers datagram from one node to physically adjacent node over a link.

Link layer services

① framing, link access → datagram to frame (+ header + trailer)
↕ channel access if medium is shared
MAC addresses in frame headers.

② reliable delivery between adjacent nodes.

③ flow control between adjacent nodes.

④ error detection ⑤ error correction without retransmission

⑥ half-duplex & full duplex
both but not the same time — both at the same time.

Host link layer implementation

- in every host. - on chip or NIC - attaches to host's system buses
- combination of hardware, software, frameware.

* EDC : error detection & correction bits. that are added to data.

D	EDC

↳ the larger the better

* error detection is not 100% reliable.

Parity checking → detects single bit errors

reciever
- → compute parity of recieved d.
- → compare with recieved parity bit → if different → error detected.
- * can detect & correct error without retransmission by
  2 dimensional parity

internet checksum ⇒ detect errors (flipped bits) in transmitted segment.

- 16 bit integer of contents of UDP header.
- compute checksum by one's complement sum.
- put value in UDP checksum field. & send.
- at reciever compute checksum for recieved data & compare with recieved checksum.

Cyclic redundancy check (CRC) → more powerful error detection.

D: data G: r+1 bits (generator)

$\langle D, R \rangle = D \cdot 2^r \oplus R$ ⇒ bits sent.

sender
1. r = n of bits in G − 1
2. add 0s as n of r to D
3. G ⟌ D
4. D ⊕ R → sent checksum
5. G ⟌ D
6. if R = 0 ✓ else error detected

MAC Addresses

IP	MAC (LAN, Physical, Ethernet addresses)
- network layer	- Link layer
- for forwarding	- to locally get frame from a physically connected interface to another (same subnet)
- 32 bit normally	- 48 bit burned in NIC ROM & sometimes software settable.

* MAC Address allocation administrated by IEEE

ARP (address resolution protocol)

↳ to determine an interface's MAC address from its IP & vice versa.
↳ each IP node on LAN has an ARP table

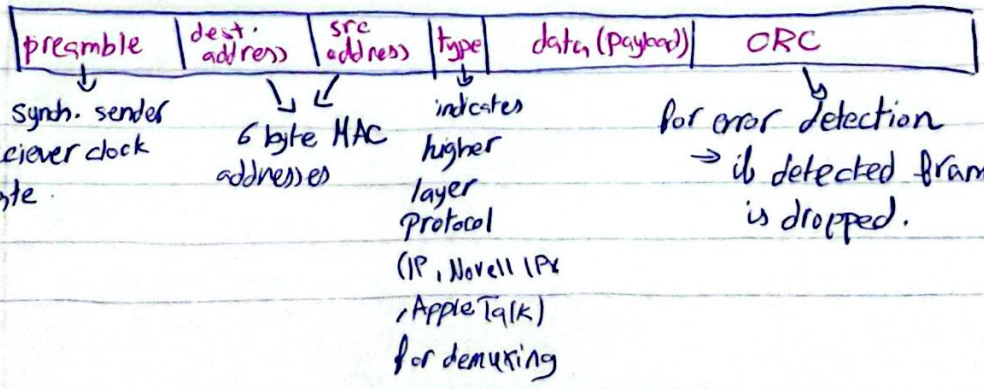
<IP, MAC, TTL>
↳ time after which this mapping will be forgotten
↳ normally 20 min

* in ARP Query ⇒ destination MAC FF-FF-FF-FF-FF-FF
↳ if same subnet → IP: target IP
↳ if not destination MAC is router's port.

Ethernet

↳ dominant wired LAN technology (10 Mbps - 400 Gbps)
↳ simple, cheap, single chip

* physical topology
- bus: in 90s (nodes can collide)
- switched: switches (no collision)



Cons :

- connectionless => no Handshaking between NICs.
- unreliable => no Acks or NAks
  - data in dropped frames can be recovered if sender uses higher layer rdt (TCP)
- ethernet's MAC Protocol is unslotted CSMA/CD with binary backoff.

Switches

- link layer device
- store, forward Ethernet or other frames
- transparent => hosts unaware of its presence
- plug & play, self learning (no configuration needed)
- ethernet protocol on each link => no collisions
  - full duplex
  - each link has its collision domain

* each switch has a **Switch Table**

[MAC of host, interface to reach host, time stamp]

* forwarding => ① get incoming link & MAC of sender
② record or look for it in table by MAC Destination
③ - if destination is same as sender drop frame
- else forward frame to destination.
- else flood

Switches Vs routers

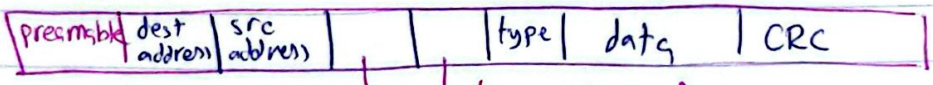
- link layer → routing algo.
- network layer → flooding, learning, MAC Addresses

VLANs

* multiple VLANs can be configured over single physical LAN.
* port based VLAN: switch ports grouped so that single physical switch operates as multiple virtual switches
  - frames in VLAN ports can only reach each other
  - dynamic membership: ports can be dynamically assigned among VLANs.
  - forwarding between VLANs is done via routing
* trunk port: carries frames between VLANs defined over multiple physical switches.

802.1Q VLAN frame format

→ frames forwarded within VLANs between switches can't be 802.1 but 802.1q so it adds/removes additional header fields for frames forwarded between trunk ports.



2 byte tag protocol identifier
81-100

tag control info.
(12 bit VLAN ID, 3 bit priority field like IP TOS, 1 bit drop eligible indicator)